Псевдоэлементы

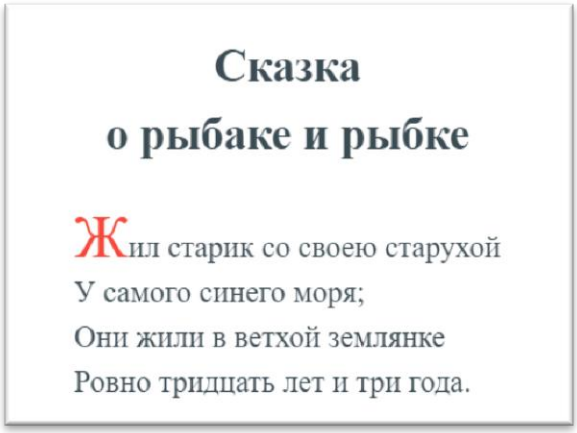
В прошлых уроках мы разобрали псевдоклассы — изменение *существующих* элементов в зависимости от их состояния. А возможно ли задать или создать стили для таких элементов, которых нет внутри HTML? Да! И для этого в CSS существует понятие псевдоэлементов.

Разберёмся на примере стилизации буквицы. Буквица — увеличенный первый символ в параграфе. Вы часто можете встретить этот приём в книгах со сказками или книгах, стилизованных под старину. Каким способом это можно сделать? Первое, что приходит в голову — обернуть первый символ в отдельный тег и стилизовать именно его.

```
<article>
  <p>
    <span class="first-letter">Ж</span>ил старик со своею старухой <br>
    У самого синего моря; <br>
    Они жили в ветхой землянке <br>
    Ровно тридцать лет и три года.
  </p>
</article>
article {
  color: #37474f;
  font: 25px/1.5 serif;
}

.first-letter {
  color: #f44336;
  font-size: 2em;
  line-height: 1;
}
```

Результат



Хороший и рабочий вариант, подходящий для точечных изменений в некоторых текстах. Какие проблемы тут могут быть скрыты? Во-первых: если таких текстов много, то добавлять теги к каждому нужному параграфу достаточно долго и риск ошибки возрастает. Можно забыть проставить тег или проставить не на первой букве. Во-вторых, масштабируемость падает. В случае избавления от буквицы нужно удалить все классы или удалить стили, но тогда останутся ненужные теги, которые скорее запутают.

Как можно выйти из этой ситуации? Тут в дело и вступают псевдоэлементы. Они могут *виртуально* создать за нас такой тег и стилизовать его используя только CSS. Это решает сразу две проблемы, которые были описаны выше:

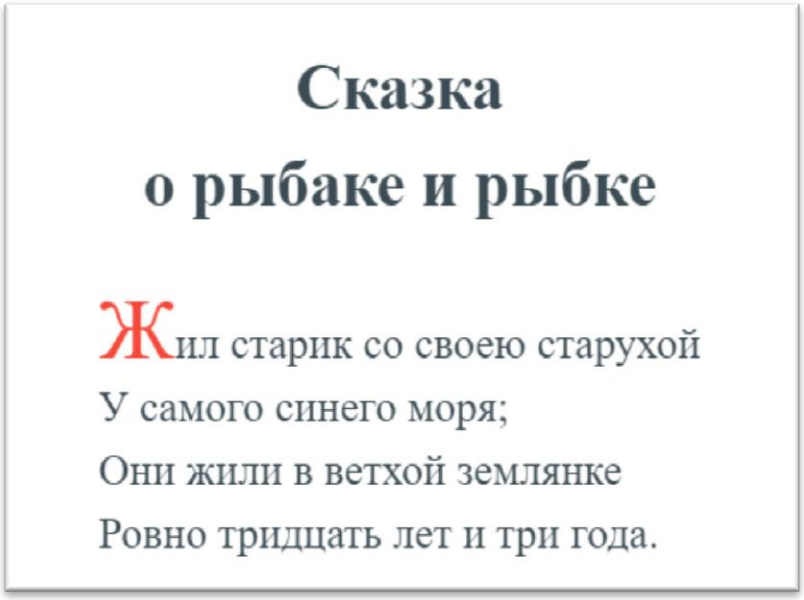
1. Нет необходимости проставлять новые теги. Достаточно только указать нужный селектор в CSS.
2. После удаления такого класса в HTML не останется ненужных тегов.

За стилизацию первого символа отвечает псевдоэлемент `::first-letter`. Он *виртуально* обернёт первый символ и применит к нему пользовательские стили. Немного перепишем пример и для всех параграфов внутри article укажем буквицу.

```
article {
  color: #37474f;
  font: 25px/1.5 serif;
}

article p::first-letter {
  color: #f44336;
  font-size: 2em;
  line-height: 1;
}
```

Результат



Результат не изменился, но, с точки зрения работы браузера, произошло много изменений. Браузер автоматически нашёл первый символ в параграфе, который находится внутри article. *Обернул* его и применил стили, описанные в CSS.

Интересно: псевдоэлементы, по синтаксису, очень похожи на псевдоклассы, но вместо одного символа `:` используется два. При этом браузеры правильно обработают такой псевдоэлемент `:first-letter`. При такой записи не сразу очевидно где псевдокласс, а где псевдоэлемент, поэтому всегда используйте `::` для указания псевдоэлемента.

before и after

Для этих двух псевдоэлементов можно написать не только урок, но и целый курс. Их взаимодействие с сайтами невозможно переоценить. Используясь в большинстве случаев, `::before` и `::after` по праву являются самыми важными псевдоэлементами. В процессе изучения вёрстки можно придумать различные способы их использования и стилизации.

В уроке про списки было сказано, что стилизация маркеров чаще всего происходит с помощью псевдоэлементов. В этой части рассмотрим, как же это делается.

Важно: здесь будут свойства, которые относятся к позиционированию элементов. Все эти свойства будут изучены в курсе CSS: Позиционирование. Если некоторые части CSS будут непонятны, то не переживайте. В скором времени вы изучите работу всех новых свойств.

`::before` и `::after` позволяют создать новый контент внутри HTML дерева. Этот контент привязан к определённому элементу и может появляться до него или после. Именно поэтому псевдоэлементы называются так:

- `::before` — псевдоэлемент, позволяющий добавить контент перед выбранным элементом.
- `::after` — псевдоэлемент, позволяющий добавить контент после выбранного элемента.

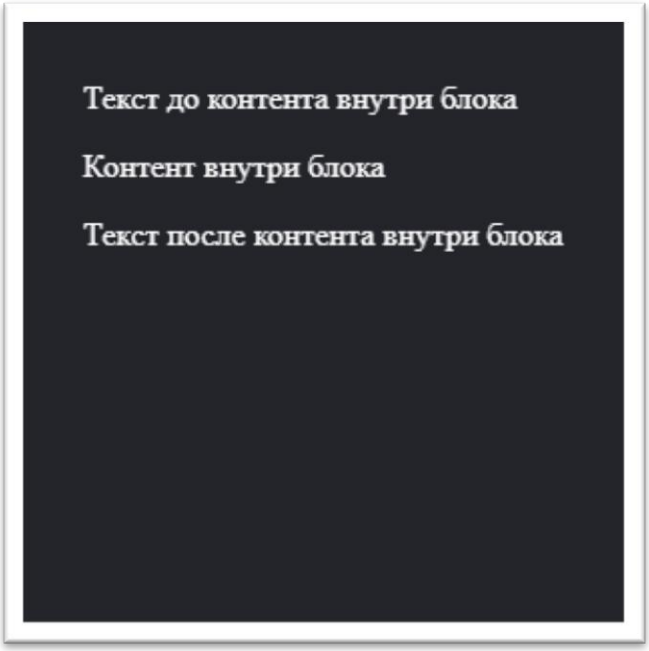
Каждый из этих псевдоэлементов должен включать специальное свойство `content`, внутри которого и указывается, что должно быть внутри. Без этого свойства псевдоэлементы `::before` и `::after` работать не будут!

Создадим блочный элемент и воспользуемся новыми псевдоэлементами.

```
<div class="square bg-black text-white">
  <p>Контент внутри блока</p>
</div>
.square::before {
  content: "Текст до контента внутри блока";
}

.square::after {
  content: "Текст после контента внутри блока";
}
```

Результат



Весь контент внутри псевдоэлементов `::before` и `::after` является строчным. То есть имеет свойство `display: inline` по умолчанию. Вы легко можете изменять это и работать с псевдоэлементами так, как если бы это были обычные элементы внутри HTML документа.

Это открывает поистине безграничные возможности стилизации элементов с помощью CSS. Много техник построено на использовании псевдоэлементов. Создадим пользовательские маркеры списка, используя изображение.

```
<h1>Обучение на Хекслете</h1>
<ul class="hexlet-ul">
  <li>Большое количество теории</li>
  <li>Множество практик и дополнительных заданий</li>
  <li>Комплексные проекты для закрепления знаний</li>
</ul>
.hexlet-ul {
  list-style: none;
}

.hexlet-ul li {
  position: relative;

  margin-bottom: 20px;
}

.hexlet-ul li::before {
  position: absolute;
  left: -30px;
```

```
  width: 20px;
  height: 20px;

  background-image: url(https://assets.codepen.io/1425525/hexlet_logo.png);
  background-repeat: no-repeat;
  background-size: cover;

  content: '';
}
```

Результат

Разберём, что происходит в псевдоэлементе `.hexlet-ul li::before`:

```
.hexlet-ul li::before {
  width: 20px;
  height: 20px;

  content: '';
}
```

В этой части CSS кода указывается пустой `content`. Так как мы хотим добавить только изображение, то никакие символы нам не нужны. Данное поле можно оставить пустым, но добавлять его обязательно. Для элемента задаются определённые рамки высоты и ширины. Именно в эти рамки мы и будем вписывать изображение.

```
.hexlet-ul li::before {
  background-image: url(https://assets.codepen.io/1425525/hexlet_logo.png);
  background-repeat: no-repeat;
  background-size: cover;
}
```

Устанавливаем изображение для блока. Для этого мы используем три свойства:

1. `background-image` — свойство, позволяющее установить изображение в качестве фона. Внутри используем функцию `url` для указания адреса, по которому расположено изображение.
2. `background-repeat` — повтор изображения. Нужно ли повторять изображение, если это возможно? В данном случае нет, так как нам нужно конкретное изображение, а не повторяющийся фон. Значение `no-repeat` запрещает повторять изображение.
3. `background-size` — размер изображения. Ключевое слово `cover` масштабирует изображение с сохранением пропорций и вписывает его в высоту или ширину существующего блока.

Позиционировав этот псевдоэлемент мы получили изображение в качестве маркера списка. Для этого не пришлось подстраивать изображение, так как эта работа осталась за CSS.

Дополнительное задание

Повторите все примеры, представленные в уроке. Попробуйте различные значения свойств. Для списка используйте псевдоэлемент `::after`, с помощью которого добавьте небольшое изображение после пункта списка.